

1237-00

U.S. APPLICATION NO. (IF KNOWN, SEE 37 CFR

09/701154

PRIORITY DATE CLAIMED
01 OCT 1999

TRANSMITTAL LETTER TO THE UNITED STATES
DESIGNATED/ELECTED OFFICE (DO/EO/US)
NOV 22 2000 CONCERNING A FILING UNDER 35 U.S.C. 371

INTERNATIONAL APPLICATION NO.
PCTUS00/26858INTERNATIONAL FILING DATE
29 SEP 2000

METHOD AND SYSTEM FOR PROVIDING DATA SECURING USING FILE SPOOFING

APPLICANT(S) FOR DO/EO/US

FRIEDMAN, George; STAREK, Robert Phillip; MURDOCK, Carlos A.

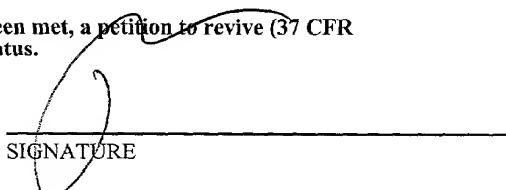
Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information:

1. This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.
2. This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.
3. This is an express request to begin national examination procedures (35 U.S.C. 371(f)) at any time rather than delay examination until the expiration of the applicable time limit set in 35 U.S.C. 371(b) and PCT Articles 22 and 39(1).
4. A proper Demand for International Preliminary Examination was made by the 19th month from the earliest claimed priority date.
5. A copy of the International Application as filed (35 U.S.C. 371 (c) (2))
 - a. is transmitted herewith (required only if not transmitted by the International Bureau).
 - b. has been transmitted by the International Bureau.
 - c. is not required, as the application was filed in the United States Receiving Office (RO/US).
6. A translation of the International Application into English (35 U.S.C. 371(c)(2)).
- A copy of the International Search Report (PCT/ISA/210).
- Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371 (c)(3))
 - a. are transmitted herewith (required only if not transmitted by the International Bureau).
 - b. have been transmitted by the International Bureau.
 - c. have not been made; however, the time limit for making such amendments has NOT expired.
 - d. have not been made and will not be made.
- A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3)).
- An oath or declaration of the inventor(s) (35 U.S.C. 371 (c)(4)).
- A copy of the International Preliminary Examination Report (PCT/IPEA/409).
- A translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371 (c)(5)).

Items 13 to 20 below concern document(s) or information included:

13. An Information Disclosure Statement under 37 CFR 1.97 and 1.98.
14. An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included.
15. A **FIRST** preliminary amendment.
16. A **SECOND** or **SUBSEQUENT** preliminary amendment.
17. A substitute specification.
18. A change of power of attorney and/or address letter.
19. Certificate of Mailing by Express Mail
20. Other items or information:

acknowledgement postcard

U.S. APPLICATION NO. (IF KNOWN, SEE 37 CFR 09/701154	INTERNATIONAL APPLICATION NO. PCT/US00/26858	ATTORNEY'S DOCKET NUMBER 1237-00		
21. The following fees are submitted:		CALCULATIONS PTO USE ONLY		
BASIC NATIONAL FEE (37 CFR 1.492 (a) (1) - (5)) :				
<input type="checkbox"/> Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1.445(a)(2) paid to USPTO and International Search Report not prepared by the EPO or JPO \$1,000.00 <input type="checkbox"/> International preliminary examination fee (37 CFR 1.482) not paid to USPTO but International Search Report prepared by the EPO or JPO \$860.00 <input checked="" type="checkbox"/> International preliminary examination fee (37 CFR 1.482) not paid to USPTO but international search fee (37 CFR 1.445(a)(2)) paid to USPTO \$710.00 <input type="checkbox"/> International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provisions of PCT Article 33(1)-(4) \$690.00 <input type="checkbox"/> International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(1)-(4) \$100.00				
ENTER APPROPRIATE BASIC FEE AMOUNT =		\$710.00		
Surcharge of \$130.00 for furnishing the oath or declaration later than months from the earliest claimed priority date (37 CFR 1.492 (e)).		<input type="checkbox"/> 20 <input type="checkbox"/> 30 \$0.00		
CLAIMS		NUMBER FILED	NUMBER EXTRA	RATE
Total claims		34 - 20 =	14	<input checked="" type="checkbox"/> \$18.00 \$252.00
Independent claims		4 - 3 =	1	<input checked="" type="checkbox"/> \$80.00 \$80.00
Multiple Dependent Claims (check if applicable).			<input type="checkbox"/>	\$0.00
TOTAL OF ABOVE CALCULATIONS =		\$1,042.00		
Reduction of 1/2 for filing by small entity, if applicable. Verified Small Entity Statement must also be filed (Note 37 CFR 1.9, 1.27, 1.28) (check if applicable).		<input type="checkbox"/> \$0.00		
SUBTOTAL =		\$1,042.00		
Processing fee of \$130.00 for furnishing the English translation later than months from the earliest claimed priority date (37 CFR 1.492 (f)).		<input type="checkbox"/> 20 <input type="checkbox"/> 30	+	\$0.00
TOTAL NATIONAL FEE =		\$1,042.00		
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31) (check if applicable).		<input checked="" type="checkbox"/> \$40.00		
TOTAL FEES ENCLOSED =		\$1,082.00		
				Amount to be: \$
				refunded
				charged
<input checked="" type="checkbox"/> A check in the amount of \$1,082.00 to cover the above fees is enclosed. <input type="checkbox"/> Please charge my Deposit Account No. 13-3405 in the amount of \$ to cover the above fees. A duplicate copy of this sheet is enclosed. <input checked="" type="checkbox"/> The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. 13-3405 A duplicate copy of this sheet is enclosed.				
NOTE: Where an appropriate time limit under 37 CFR 1.494 or 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed and granted to restore the application to pending status.				
SEND ALL CORRESPONDENCE TO: PAUL A. TAUFER, ESQ. SCHNADER HARRISON SEGAL & LEWIS, LLP 1600 MARKET STREET, SUITE 3600 PHILADELPHIA, PA 19103 (215) 751-2476 (215) 568-6946 - FAX				
 SIGNATURE Paul A. Taufer NAME 35,703 REGISTRATION NUMBER 11/22/00 DATE				

09/701154

METHOD AND SYSTEM FOR PROVIDING DATA SECURITY
USING FILE SPOOFING

Field of the Invention

5 The present invention pertains to the field of file systems in electronic computers. In particular, the invention relates to a method and system for providing data security using file spoofing.

Background of the Invention

10 Data security is a serious concern of computer users and owners of intellectual property. It is increasingly common to use measures such as encryption to secure data files, to protect data from loss or unauthorized activity.

15 Computer systems typically include one or more local or networked data storage devices. A typical application program executing on such a computer system accesses such data storage devices by calling standard file system services provided by an operating system, such as services for creating, reading, and writing files on the data storage devices.

20 A device driver is a set of computer-implemented instructions that implements the device-specific aspects of generic input/output operations. In typical operating systems, software applications such as device drivers run in either "kernel mode" or "user mode." A virtual device driver is a type of device driver that has direct access to an operating system kernel, such as by running in kernel mode. "Kernel mode" is a highly privileged memory access mode of the processor. "User mode" is a less privileged memory access mode of the processor. The memory access mode is a part of the hardware state of the processor. The kernel mode privilege level is also known as "Ring 0," and the user mode privilege level is also known as "Ring 3." Kernel mode access allows the virtual device driver to interact with system and hardware resources at a very low level.

25 In conventional operating systems, device drivers may be represented as layered on top of one another. The layered architecture is also sometimes referred to as a stack or a calling chain. It is the lowest-level device driver that typically controls a hardware device. If there is only a single device driver above the hardware device, the driver is called a monolithic driver. However, a plurality of drivers may be placed above the lowest-level driver. Input and output requests ("I/O requests") to the hardware device or devices controlled by a lowest-level driver are handled first by the highest-level driver, then seriatim by any lower-level intermediate drivers,

and finally by the lowest-level driver.

A file system driver is generally a highest-level driver, layered above a device driver for a data storage device such as a hard disk drive. The file system driver implements high-level aspects of I/O requests directed to the file system, such as requests to create, open, extend, and 5 delete files and directories. A plurality of file system drivers may exist in a single computer, and file system drivers may be specific to different types of file systems, such as the FAT and NTFS file systems.

It is known in the art to monitor file I/O requests in operating systems having an installable file system manager and layered device drivers, such as the Windows 95®, Windows 10 98®, and Windows Me® operating systems available from Microsoft Corporation of Redmond, Washington, and collectively referred to herein as "Windows 9x". In Windows 9x operating systems, file system monitoring may be accomplished by registering a file system applications programming interface hook with the installable file system manager. Windows 9x provides a function called IFSMGR_InstallFileSystemApiHook which is designed to be used for monitoring 15 I/O requests to a file system. This service allows virtual device drivers to monitor all file system activity by hooking into the file system calls. By means of a call during system initialization to IFSMGR_InstallFileSystemApiHook, a virtual device driver may insert itself onto the stack of all file system requests.

A somewhat different approach has been used to monitor file systems on object-oriented 20 operating systems, such as the Windows NT® operating system and successor operating systems such as Windows 2000®, available from Microsoft Corporation of Redmond, Washington, and collectively referred to herein as "Windows NT." In Windows NT, I/O requests are described by data structures known as I/O Request Packets ("IRPs"), which are used for communication 25 between software applications and drivers. All IRPs to hardware devices are handled by device drivers operating in kernel mode. High-level, intermediate, and low-level drivers exchange IRPs to complete a given I/O request. The lowest-level driver calls an NT layer known as the Hardware Access Layer (HAL) to gain direct control of the hardware. It is known on a Windows NT system to implement a file system monitor as a device driver object that creates filter device objects and attaches those objects to target file system device objects, so that the file system 30 monitor will see all IRPs directed to the monitored data storage devices.

It is known to store secured files, such as encrypted files, alongside unsecured files in the same file system. The encrypted file appears in the file directory like any other file, with relevant file attributes such as name and size. However, the data contained in the file is unintelligible to

user applications until decrypted. Furthermore, the encryption process is likely to result in the size of the file becoming larger or smaller than the original unencrypted data. In such a case, a request to the file system to determine the file size would not reliably return the actual size of the original data. From the user's point of view, this type of data security lacks the desirable feature 5 of transparency.

It is also known to store secured files in a special physical or virtual location apart from the ordinary file system. Such locations may include remote networked devices, encrypted or password-protected file systems, or other virtual secured file systems. This type of data security prevents the user from freely intermingling secured and unsecured files in a single file directory, 10 even though the files may be logically related to one another. Although a user may set up, in the unsecured directory, symbolic links or shortcuts to secured files in another location, such an exercise for authorized persons adds an undesirable layer of obfuscation and effort to the process of conveniently accessing secured data.

15 **Summary of the Invention**

It is a principal object of the present invention to provide a more convenient way for users to obtain information in connection with secured data files or file systems.

Another object of the present invention is to prevent unauthorized device drivers from obtaining information in connection with secured data files or file systems.

20 These and other objects are provided by a method and system for providing data security using file spoofing.

More particularly, the present invention relates to a method for providing data security in a device driver for accessing data. The device driver detects a file system request, completes the file system request, and receives return information from the file system request. The device 25 driver further determines whether the file system request is for a tag file associated with a secured file; and if so, modifies the return information to reflect a file attribute of the secured file.

In another aspect, the invention relates to a system for providing data security, the system comprising a device driver for accessing data. The device driver is operably installed in an 30 operating system on an electronic computer. The device driver detects a file system request, completes the file system request, receives return information from the file system request, determines whether the file system request is for a tag file associated with a secured file; and if so, modifies the return information to reflect a file attribute of the secured file.

In still another aspect, the invention comprises a machine-readable medium comprising a device driver program for accessing data. In yet another aspect, the invention comprises a computer-implemented device driver for accessing data.

Further objects and advantages of this invention will become apparent from the detailed 5 description of a preferred embodiment, which follows.

Brief Description of the Drawings

The present invention is described in further detail with reference to the accompanying drawings. The figures of the accompanying drawings illustrate the present invention by way of 10 example and not limitation.

FIG. 1 is a diagram of the system architecture layout of the Windows 9x operating system.

FIG. 2 is a diagram of the system architecture layout of the Windows NT operating system.

15 FIG. 3 is a flow chart illustrating an embodiment of the file spoofing method of the present invention.

Detailed Description of Preferred Embodiments of the Invention

The invention will be understood more fully from the detailed description given below; 20 which, however, should not be taken to limit the invention to a specific embodiment, but is for explanation and understanding only.

The terms "computer" or "computer system," as used herein, include any device capable of receiving, transmitting, and/or using information, including, without limitation, a processor; a microprocessor; a personal computer, such as a laptop, palm PC, desktop or workstation; a 25 network server; a mainframe; an electronic wired or wireless device, such as for example, a telephone; an interactive television or electronic box attached to a television, such as for example, a television adapted to be connected to the Internet; a cellular telephone; a personal digital assistant; an electronic pager; and a digital watch. In an illustrative example, information is transmitted in the form of e-mail. A computer, computer system, or system of the invention 30 may operate in communication with other systems over a network, such as, for example, the Internet, an intranet, or an extranet, or may operate as a stand-alone system.

It should also be understood that the terms "device driver" or "driver," as used herein, include any computer-implemented instructions for directly or indirectly accessing or controlling

hardware devices, including, without limitation, device drivers, virtual device drivers (VxDs), instructions using NT kernel mode architecture, instructions using Win32 driver model (WDM), and other instructions, in any computer language, directed to any computer, computer architecture, network, or operating system.

5 The terms "information" and "data" as used herein are each intended to include the broadest definition of the other, and each include text, audio and video data. By way of further example, the term "information" can mean raw data, processed data, or a combination of raw and processed data.

10 Although the embodiment illustrated in the figures comprises a device driver described for illustrative purposes as a "file system monitor," the term "file system monitor" as used herein refers generally to a device driver of any kind using the file spoofing of the present invention. Device drivers within the scope of the invention may perform any sort of useful function that may be performed by a device driver, including, without limitation, general-purpose monitoring, permission monitoring, filtering, encryption, decryption, virus detection, data mirroring, I/O 15 functions directed toward any device, and other functions, and are not limited to either monitoring or to functions related to file systems. Any device driver which accomplishes file spoofing appropriately falls within the scope of the present invention.

15 One embodiment of the present invention may be implemented on a Windows 9x operating system. Referring now to FIG. 1, components of the Windows 9x operating system are 20 divided between user mode code 10 and kernel mode code 30, which provide different levels of system protection. For one embodiment, the user mode code 10 includes a system virtual machine 20 capable of running 16-bit and 32-bit software applications 21-22, and a plurality of MS-DOS virtual machines 25. In this embodiment, the kernel mode code 30 comprises low-level operating system services and virtual device drivers, such as a virtual machine manager 40, 25 a file system monitor 50 of the present invention, and an installable file system manager 60.

20 Beneath the installable file system manager 60 are a plurality of file system drivers 70-72 for file systems such as FAT and NTFS. Beneath the file system drivers 70-72 is a block I/O subsystem 80. The block I/O subsystem 80 includes an I/O supervisor 81 which manages requests as they pass through the file system hierarchy, a monolithic driver 82 for port I/O, and a 30 layered plurality of device drivers 83-84.

In this embodiment, the first device driver 50 intercepts all I/O requests from user mode code 10 and from applications 21-22 running in user mode 10, before the I/O requests are sent to the installable file system manager 60. The first device driver 50 is able to monitor and, if

desired, filter all file system activity occurring in the installable file system manager 60, file system drivers 70-72, and block I/O subsystem 80. By means of a call during system initialization to IFSMGR_InstallFileSystemApiHook, the first device driver 50 is hooked into such calls when the operating system is started or restarted, at which time it is inserted into a

5 functionally uppermost position on the stack of all file system requests. From the installable file system manager 60 down through each driver in the layered plurality 83-84, an I/O request is passed from the highest level to the lowest level, and the devices can also view the result of a request as it passes back up the stack to the source of the I/O request. Each device driver on the stack may service an I/O request itself and not pass the I/O request to lower levels, or may, if

10 desired, itself generate a new I/O request. Such device drivers may implement functions that require waiting, such as for an interrupt, or for a device to become available. During such waiting periods the device driver simply returns to its caller, allowing the calling application or device driver to perform other work in parallel with the I/O request. In the alternative, the calling application or device driver can simply wait ("block") until the I/O request is complete.

15 In another embodiment, illustrated with reference to FIG. 2, the present invention may be implemented on a Windows NT operating system. As is well known in the art, an application 100 running in user mode under Windows NT may send an I/O request to operating system services 110. I/O manager 120 receives I/O requests, and coordinates the transfer of I/O request packets among various drivers. In the alternative, the various drivers may communicate directly 20 with each other without using an I/O manager 120 or other device to coordinate transfer of information among the various drivers.

The conventional input/output system of operating systems such as Windows NT comprises a plurality of device drivers 130-132 for processing I/O requests. Such device drivers are illustrated, for example, by file system drivers 130, and a layered plurality of device drivers 25 131-132. The I/O manager 120 typically delivers I/O request packets to the file system driver 130 responsible for managing the target of the I/O request. However, as is known in the art, a file system monitor 50 can attach to other device drivers 130-132 in an object-oriented fashion. Thereupon, the I/O manager 120 routes I/O request packets intended for the target device driver 130-132 to the file system monitor 50 that has attached to the target device driver 130-132. In 30 this illustrative embodiment, the file system monitor 50 attaches to each of the plurality of file system driver objects 130.

FIG. 3 is a flow chart of one embodiment of a method for providing data security in a file system monitor 50 using file spoofing.

As shown in FIG. 3, the file spoofing process of the present invention is initiated, in step 200, each time a file system request is detected. In step 210, the process determines whether or not the file system request involves a spoofed file. The determination of step 210 is performed for any file system request that specifies a named file as the object of the request. Such file system requests include FILE_OPEN to open a file, FILE_DELETE to delete a file, and FILE_RENAME to rename a file. In the Windows NT operating systems, such file system requests also include FILE_QUERY_INFORMATION to query file information, and FILE_SET_INFORMATION to set file information. Each of these calls requires a file name to be specified. In the determination of step 210, the specified file name is checked to determine whether the file is a spoofed file.

A spoofed file is a file which has been secured by placing the data in a secured file location where such data is not readily accessible to the user, such as, by way of illustration and not limitation, a secure or encrypted virtual file system, while maintaining a tag file in a user-accessible part of the file system to serve as a placeholder. To save disk space, the tag file may be zero bytes in length. In one embodiment, from the point of view of the user, the tag file transparently appears to contain the secured data, and the file spoofing process will enable any file system request to reach the secured file location.

To determine whether a file is a spoofed file, the process checks the file name against a database of all spoofed files, and if the file is a spoofed file, the process determines the secured file associated with the tag file. In an alternate embodiment, the process may rely on data stored in the tag file.

If the determination of step 210 is that the file request involves a spoofed file, the process continues at step 211 and completes the file system request. In one embodiment, the process calls the next lower driver to complete the file system request for the tag file. In an alternate embodiment, the process rewrites the file system request to refer to the secured file instead of the tag file, and calls the next lower driver to complete the file system request for the secured file.

The process continues at step 212 by modifying the information returned by the file system request. File attributes, such as file size, are part of the information returned by file system requests. The process removes selected file attributes of the tag file from the return information, and substitutes the corresponding file attributes of the secured file. For example, where the file size of a tag file is zero, the user will instead see the file size of the corresponding secured file.

The process then concludes at step 240.

Returning to step 210, if the determination of step 210 is that the file request does not involve a spoofed file, the process continues at step 220 and determines whether the file system request involves a directory call that may return a spoofed file. Such file system requests in Windows 9x include FIND_OPEN or FIND_FIRST to find a first matching file, and

5 FIND_NEXT to find a next matching file. In the Windows NT operating systems, such file system requests include DIRECTORY_CONTROL, which provides a buffer of matching file names.

If the determination of step 220 is that the file system request involves a directory call that may return a spoofed file, the process continues at step 221 and completes the file system

10 request. In one embodiment, the process calls the next lower driver to complete the file system request for the tag file. In an alternate embodiment, the process rewrites the file system request to refer to the secured file instead of the tag file, and calls the next lower driver to complete the file system request for the secured file.

The process continues at step 222 by determining whether the information returned by the

15 file system request refers to any spoofed files. Requests to find a first matching file or a next matching file will each return a single file. A request to the Windows NT file system for directory control will return a buffer of file names, each of which must be considered. If no spoofed files are returned, the process concludes at step 240.

If any spoofed files are returned, the process continues at step 225 by modifying the information returned by the file system request. File attributes, such as file size, are part of the information returned by file system requests. The process removes selected file attributes of the tag file from the return information, and substitutes the corresponding file attributes of the secured file. For example, where the file size of a tag file is zero, the user will instead see the file size of the corresponding secured file. The process then concludes at step 240.

25 Returning to step 220, if the determination of step 220 is that the file system request does not involve a directory call that may return a spoofed file, the process continues at step 230 and completes the file system request, and concludes at step 240.

What have been described are only some examples of methods and systems according to the invention. Various modifications to the preferred embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown. On the contrary, it is to be understood that various and numerous other arrangements may be devised by

one skilled in the art without departing from the spirit and scope of the invention as limited only by the accompanying claims.

We claim:

1. A method for providing data security in a device driver for accessing data, the method comprising the steps of:

5

detecting a file system request;

completing said file system request;

10

receiving return information from said file system request;

determining whether said file system request is for a tag file associated with a secured file; and

15

if so, modifying said return information to reflect a file attribute of the secured file.

2. The method of claim 1 wherein said file attribute is file size.

3. The method of claim 1 wherein the step of determining further comprises the steps of:

20

determining whether said return information identifies a plurality of tag files associated with a plurality of secured files; and

25 if so, modifying said return information to reflect a file attribute of the plurality of secured files.

4. The method of claim 1 wherein the secured file is stored in encrypted form.

5. The method of claim 1 wherein the secured file is stored in a secure virtual file system.

30

6. The method of claim 1 wherein the secured file is stored on a remote networked device.

7. The method of claim 1 wherein the file system request is to open a file.

8. The method of claim 1 wherein the file system request is to delete a file.

9. The method of claim 1 wherein the file system request is to rename a file.

5 10. The method of claim 1 wherein the file system request is to query file information.

11. The method of claim 1 wherein the file system request is to set file information.

12. The method of claim 3 wherein the file system request is to find a first matching file.

10 13. The method of claim 3 wherein the file system request is to find a next matching file.

14. The method of claim 3 wherein the file system request is directory control.

15 15. A system for providing data security, the system comprising a device driver for accessing data, the device driver operably installed in an operating system on an electronic computer, wherein said device driver:

20 detects a file system request;

 completes said file system request;

 receives return information from said file system request;

25 determines whether said file system request is for a tag file associated with a secured file;
and

 if so, modifies said return information to reflect a file attribute of the secured file.

30 16. The system of claim 15 wherein said file attribute is file size.

17. The system of claim 15 wherein said device driver further

determines whether said return information identifies a plurality of tag files associated with a plurality of secured files; and

if so, modifies said return information to reflect a file attribute of the plurality of secured

5 files.

18. The system of claim 15 wherein said first device driver is a file system monitor.

19. The system of claim 15 wherein the secured file is stored in encrypted form.

10

20. The system of claim 15 wherein the secured file is stored in a secure virtual file system.

21. The system of claim 15 wherein the secured file is stored on a remote networked device.

15

22. The system of claim 15 wherein the file system request is to open a file.

23. The system of claim 15 wherein the file system request is to delete a file.

20

24. The system of claim 15 wherein the file system request is to rename a file.

25. The system of claim 15 wherein the file system request is to query file information.

26. The system of claim 15 wherein the file system request is to set file information.

25

27. The system of claim 17 wherein the file system request is to find a first matching file.

28. The system of claim 17 wherein the file system request is to find a next matching file.

30

29. The system of claim 17 wherein the file system request is directory control.

30. A machine-readable medium comprising a device driver program for accessing data, said device driver program comprising:

computer-implemented instructions for detecting a file system request;

computer-implemented instructions for completing said file system request;

5 computer-implemented instructions for receiving return information from said file system request;

computer-implemented instructions for determining whether said file system request is for a tag file associated with a secured file; and

10 computer-implemented instructions for modifying said return information to reflect a file attribute of the secured file, if said file system request is for a tag file associated with a secured file.

15 31. The machine-readable medium of claim 30 wherein the device driver program further comprises:

computer-implemented instructions for determining whether said return information identifies a plurality of tag files associated with a plurality of secured files; and

20 computer-implemented instructions for modifying said return information to reflect a file attribute of the plurality of secured files, if said return information identifies a plurality of tag files associated with a plurality of secured files.

25 32. A computer-implemented device driver for accessing data when operably installed in a computer operating system, said device driver comprising:

means for detecting a file system request;

30 means for completing said file system request;

means for receiving return information from said file system request;

means for determining whether said file system request is for a tag file associated with a secured file; and

means for modifying said return information to reflect a file attribute of the secured file,

5 if said file system request is for a tag file associated with a secured file.

33. The computer-implemented device driver of claim 32 wherein said file attribute is file size.

10 34. The computer-implemented device driver of claim 32 further comprising:

means for determining whether said return information identifies a plurality of tag files associated with a plurality of secured files; and

15 means for modifying said return information to reflect a file attribute of the plurality of secured files, if said return information identifies a plurality of tag files associated with a plurality of secured files.

Abstract

A method for providing data security in a device driver for accessing data. The device driver detects a file system request, completes the file system request, and receives return information from the file system request. The device driver further determines whether the file system request is for a tag file associated with a secured file; and if so, modifies the return information to reflect a file attribute of the secured file.

09/701154

1/3

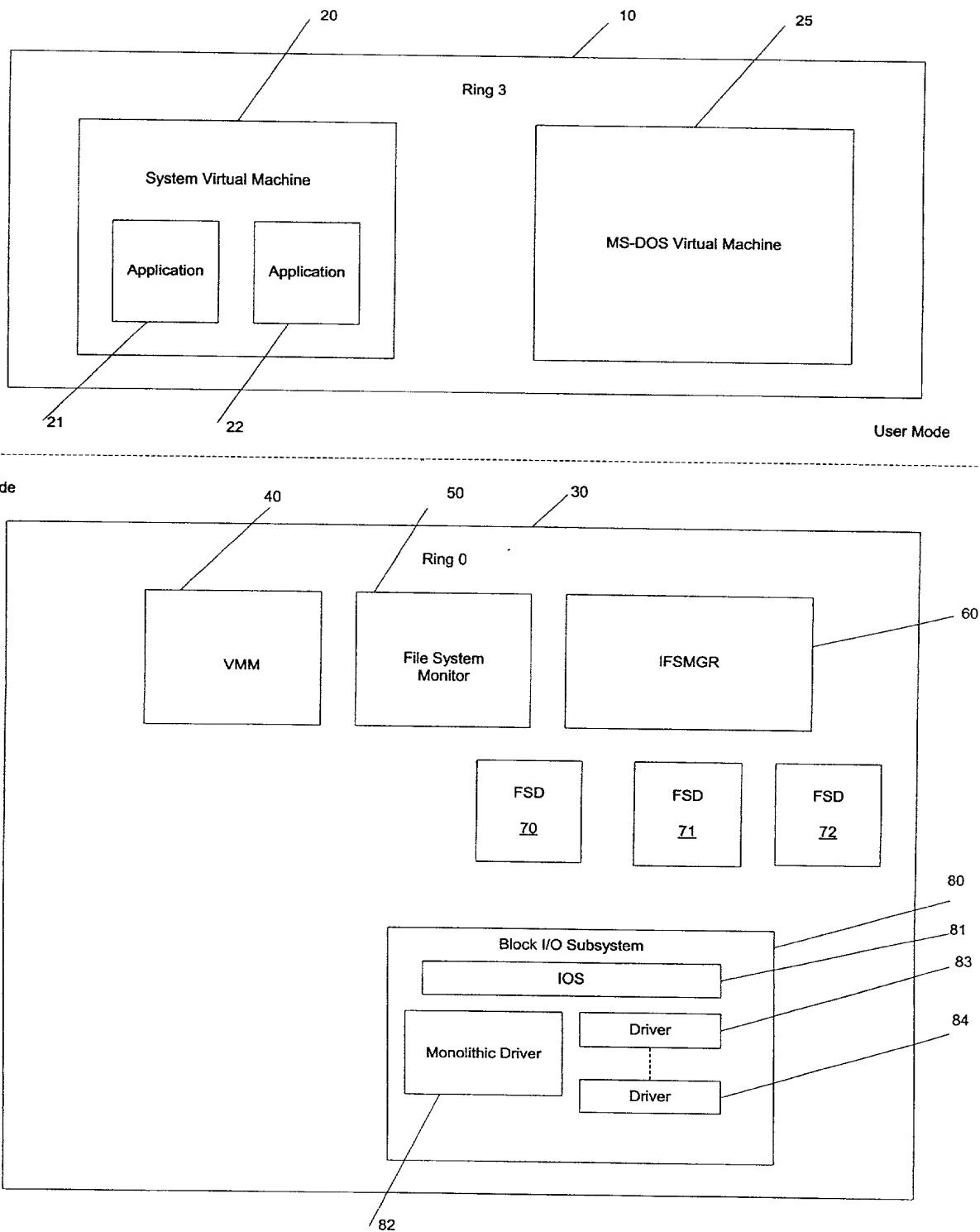


FIG. 1

09/701154

2/3

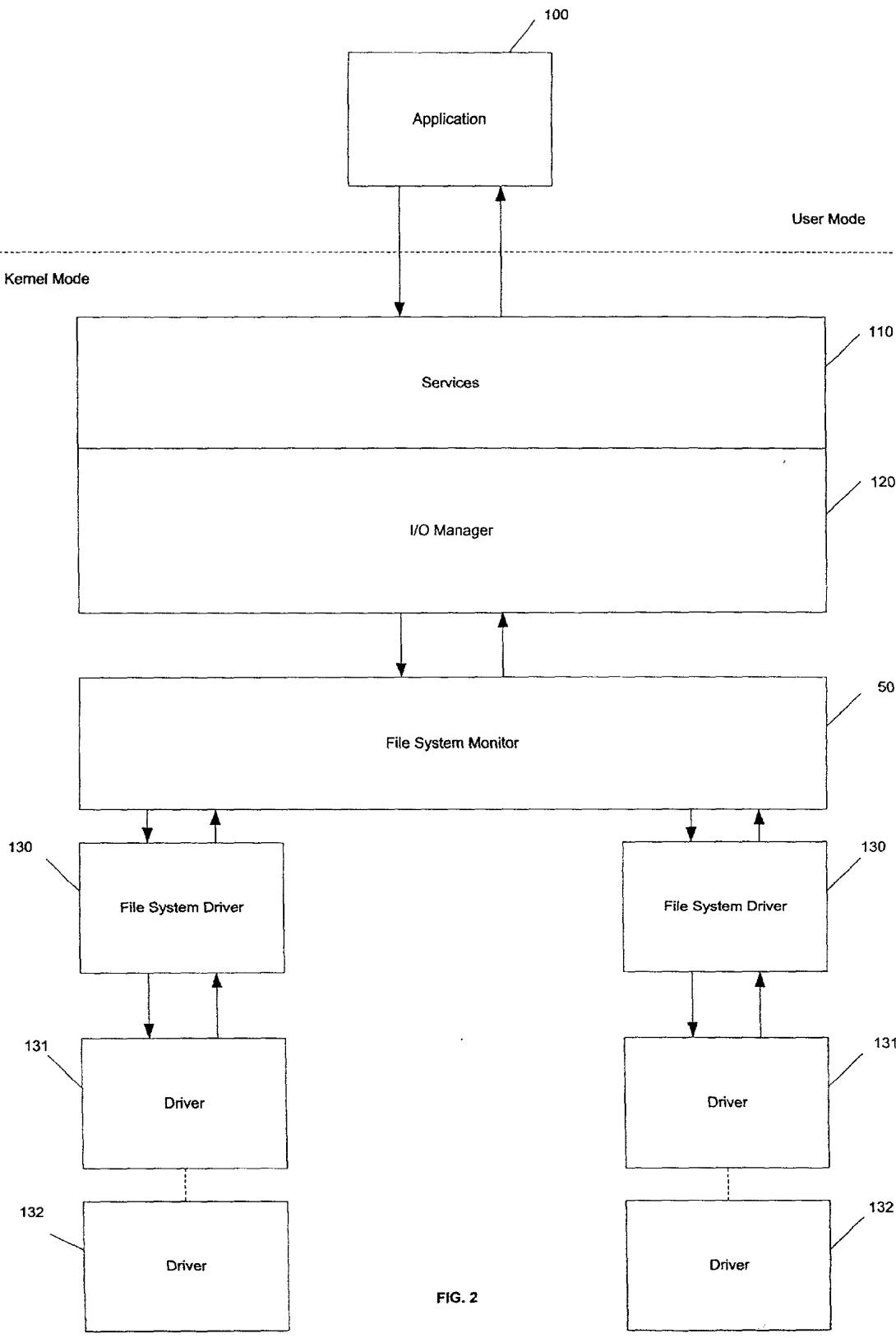


FIG. 2

09/701154

3/3

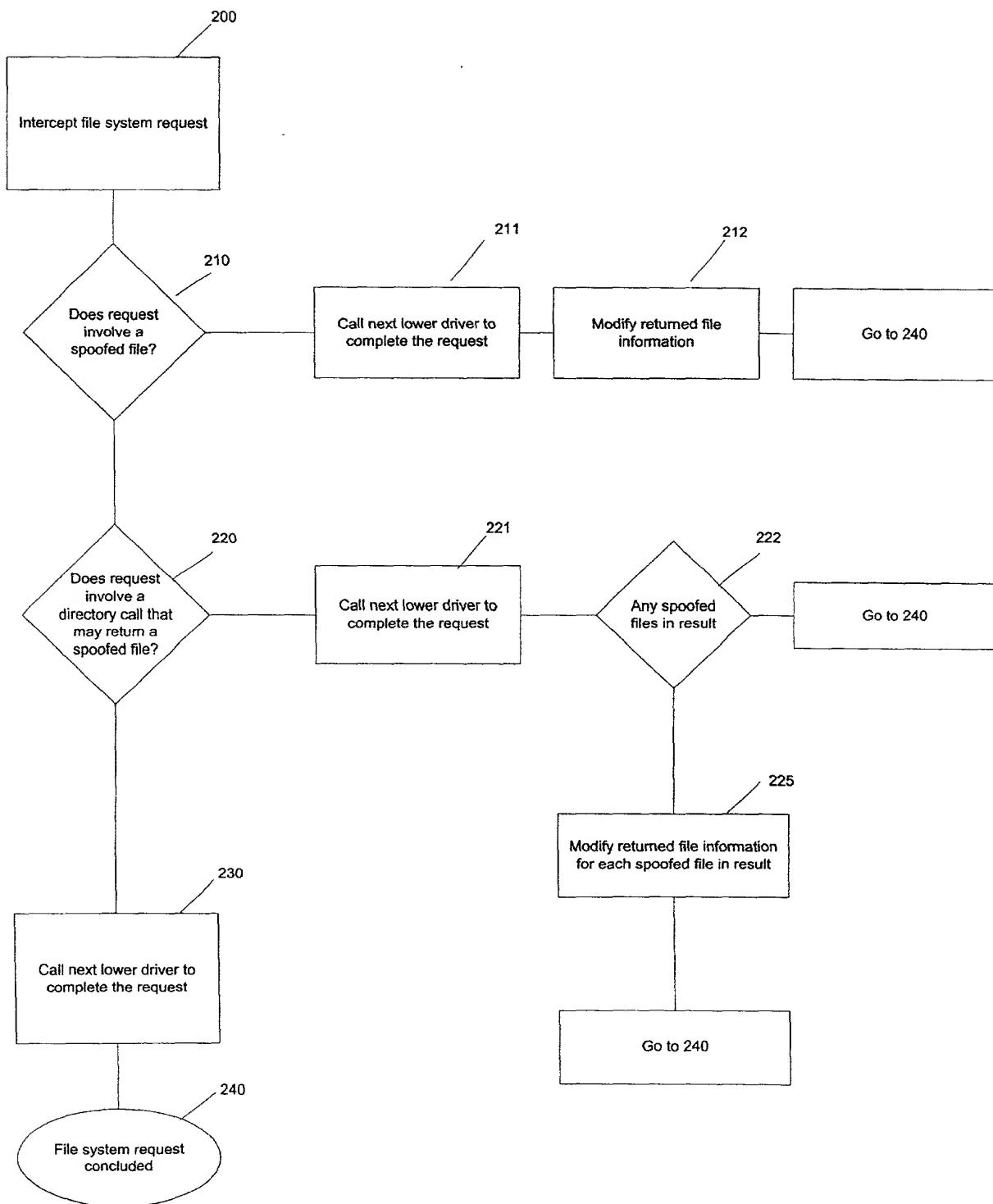
RECORDED BY COMPUTER SYSTEM
10/10/2000 036200

FIG. 3

Declaration and Power of Attorney for Patent Application

As the below named inventor, we hereby declare that:

Our residence, post office address and citizenship are as stated next to our names,

We believe we are the original and first inventors of the subject matter which is claimed and for which a patent is sought on the invention entitled **METHOD & SYSTEM FOR PROVIDING DATA SECURITY USING FILE SPOOFING** the specification of which is filed herewith.

We hereby state that we have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

We acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, §1.56(a).

We hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before the application on which priority is claimed:

Prior Foreign Application(s)	Priority Claimed			
(Number)	(Country)	(Day/Month/Year Filed)	Yes	No

We hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, we acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, §1.56(a) which occurred between the filing date of the prior application and the national or PCT international filing date of the application:

PCT/US00/26858	9/29/00	pending
(Application Serial No.)	(Filing Date)	(Status)
(patent, pending, abandoned)		

We hereby appoint the following attorneys and/or agents to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith:



T. Daniel Christenbury	Reg. No. 31,750
Guy T. Donatiello	Reg. No. 33,167
Paul A. Taufer	Reg. No. 35,703
Austin R. Miller	Reg. No. 16,602
James A. Drobile	Reg. No. 19,690
Gerard J. Weiser	Reg. No. 19,763
Robert A. McKinley	Reg. No. 43,793
Michael A. Patané	Reg. No. 42,982
Joan T. Kluger	Reg. No. 38,940
Sharon Fenick	Reg. No. 45,269
Stewart M. Wiener	Reg. No. 46,201
Armando A. Flores	Reg. No. 41,754
Felicity Rowe	Reg. No. 47,042

(13)

Address all telephone calls to Paul A. Taufer, Schnader Harrison Segal & Lewis LLP, Suite 3600, 1600 Market Street, Philadelphia, PA 19103 (215) 751-2475.

Address all correspondence to Paul A. Taufer, Schnader Harrison Segal & Lewis LLP, Suite 3600, 1600 Market Street, Philadelphia, PA 19103.

We hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under §1001 of Title 18 of the United States Code and that such wilful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of first and joint inventor: George Friedman

Inventor signature George Friedman

11/15/2000

Date

Residence: 7109 Montana Norte, Austin, Texas 78727

Citizenship: USA

Mailing Address: same as above

TX

2-00

Full name of second and joint inventor: Robert Phillip Starek

Inventor signature Robert Phillip Starek

11/15/2000

Date

U

Residence: 1807 W. Slaughter Lane #200-482, Austin, Texas 78748

Citizenship: USA

Mailing Address: same as above

TX

Full name of third and joint inventor: Carlos A. Murdock

3-0

Inventor signature Carlos A. Murdock

11/15/2002

Date

Residence: 4517 Avenue F, Austin, Texas 78751

Citizenship: USA

Mailing Address: same as above

TX

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

7